DBMS Implementation Project Phase 2: Graph Database

Group Members:

Elan Markov

Simarpreet Kaur

Prachi Sharma

Jayanth Kumar Melinavolagerehalli Jayaramaiah

Harshdeep Singh Sandhu

Priyekant Aghi

Shalmali Bhoir

CSE510: Database Management System Implementation

**Abstract**: This report details the results of Phase 1 of the CSE 510 project, where the test cases of the Java implementation of the Minibase database management system are evaluated. This system includes a series of interactive tests, which evaluate the performance of the B+ tree structure of the code in response to various types of user input, and a series of non-interactive tests, which check subsystems of the larger database management system in response to intended and erroneous input. The results of each test, along with the explanation of the function of each test and the subsystem that is being tested, are included within this document. As no changes have been made to the original implementation of the Minibase code, each test completes successfully as expected for a working implementation. However, modifications that are later made to the Minibase implementation will be evaluated against the test cases included here, so it is important to understand the function of each of these subsystems and what the tests are evaluating to understand how to properly modify the database management system to add new functionality.

Keywords: Minibase, database management system, buffer management, B+ trees, unit tests

**Introduction**

This report details the results of Phase 1 of the CSE510 final project. For this phase, the goal is to understand and describe the functionality of the tests included in the included database management system (DBMS) implementation. The database management system included in this project is Minibase, a Java implementation of a relational DBMS. In future steps, new functionality will be added to this DBMS; for Phase 1, the functionality of the system will be tested and described using the tests included within the original implementation.

The tests included within the DBMS implementation include non-interactive tests, which run automatically, and interactive tests, which require user input. The interactive tests evaluate the functionality of the binary tree implementation of the DBMS and provide the user with a menu with 20 different functions that involve operations on files and records within the database. Each set of non-interactive tests will test a certain subsystem within the DBMS; non-interactive tests include a set of tests for the buffer management system, the disk space management system, heap files, the index, the sorting functionality, and SQL queries to the DBMS. All of these tests will be described within this document; the full text of the typescript file generated from running these tests is included in Appendix A.

**Interactive Tests**

The interactive portion of the DBMS tests involves the user making queries as selected from a menu for testing the binary tree (BTree) functionality of the DBMS. However, while the desired search properties of a database lend themselves well to a tree-like structure, the storage of database files on the disk do not lend themselves well to a binary search tree. Instead, a similar structure, a B+ tree, which is similar but has nodes that store up to d-1 keys and up to d children (where d is an integer value; a BST is a special case with d=2) [1]. The interactive tests will involve making queries to files stored with a B+ tree structure. For this portion of the test, temporary files with a name of the form AAAX, where X is a nonnegative integer, are created for the duration of the test. There are 20 commands included within the interactive menu; the function of each of these is described below.

*[0] Naive delete (new file)*

*[1] Full delete(Default) (new file)*

Each of these two commands will create a new file and modify the method by which entries will be deleted in that new file. There are two algorithms for deleting files within the BTreeFile class which can be used: the naïve delete algorithm or the full delete algorithm. The naïve delete algorithm is an iterative implementation which will remove a specified data entry (as a <key, rid> tuple) from the current file without resorting, whereas the full delete algorithm is a recursive implementation that can delete specified entries and will resort the BTree.

*[2] Print the B+ Tree Structure*

This command will print the structure of the B+ tree that has been generated from all included elements. This will print a “1” and the ID of the root on the first line, then the rest of the pages of the tree in a hierarchical order. If the current file is empty, this command will return a Null Pointer Exception.

*[3] Print All Leaf Pages*

This command will each of the leaves of the BTree, i.e. all of the entries in the tree in <key, rid> tuple form. If the current file is empty, this command will return a Null Pointer Exception.

*[4] Choose a Page to Print*

This command will print the specified page.

*[5] Insert a Record*

This command will insert an integer value entry into the BTree of the current file.

[6] Delete a Record

This command will delete the entry which has the specified integer value in the current file, if it is in the file.

*[7] Test1 (new file): insert n records in order*

*[8] Test2 (new file): insert n records in reverse order*

*[9] Test3 (new file): insert n records in random order*

*[10] Test4 (new file): insert n records in random order*

*and delete m records randomly*

These command will each create a new AAAX file and insert the integers 0 to n-1, where n is an integer value specified by the user. The Test1 command will insert them in order (starting with 0 and increasing), Test2 will insert them in reverse order (starting with n-1 and decreasing), Test3 will insert them in a random order, and Test4 will insert them in a random order and then delete m random records from that tree. Note that since the tree is self-sorting, the first three tests should yield the same result, assuming the sort functionality is working properly.

*[11] Delete some records*

This command will delete any records with a key value within a specified range.

*[12] Initialize a Scan*

This command allows the user to specify an integer range of elements to be scanned (output in order as they are scanned).

*[13] Scan the next Record*

This will output the next element within the current scan within the tree.

*[14] Delete the just-scanned record*

This will delete the element last scanned within the tree.

*[15] Test5 (new file): insert n records in random order*

*and delete m records randomly.*

This has the same functionality as the previously given Test4 command, but it will use a string key instead of an integer key. As an “insert string record” function is not currently given within this element, this will merely insert integer values converted to strings as the keys for the records.

*[16] Close the file*

This command will close the currently open AAAX file, which will not delete the file but will not allow any further insert/delete/scan/print commands to be performed on it until reopened.

*[17] Open which file (input an integer for the file name):*

This will open the AAAX file specified by the integer X.

*[18] Destroy which file (input an integer for the file name):*

This will delete all entries within the AAAX file specified by the integer X.

*[19] Quit!*

This command will end the interactive portion of the tests, delete all AAAX files, and the test run will proceed to finish the remaining non-interactive tests.

**Non-Interactive Tests**

These tests are performed automatically by the program and will each test a subsystem of the DBMS, including the buffer management system, the disk space management system, heap files, the index, the sorting functionality, and SQL queries to the DBMS. The purpose of each system, the tests performed, and their results, are described in this section.

The buffer management system is responsible for allocating buffer space, which holds copies of blocks of memory for more efficient access of memory [2]. Since accessing the main memory for each query is a very expensive process in terms of time spent, this makes the usage of the DBMS much more practical and is necessary to ensure that it functions properly. In this test file, three tests for the functionality of the buffer management system are performed. The first test performs basic buffer operations: allocate a page, write then read the page, then deallocate the page. The second test performs a series of operations that are illegal which are supposed to fail: pinning too many pages, unpinning a double-pinned page, and unpinning a page not in the buffer pool. The third test checks internals of the buffer management system, including allocating and reading some pages. All tests yield the expected result and complete successfully.

The disk space management system is part of the logical layer of a DBMS responsible for allocate/deallocate and read/write operations, necessary for adding to or removing from a database [3]. Four tests are performed to ensure the proper functionality of the disk space system. The first performs tests on standard disk space operations: allocate pages, write to pages, and deallocate empty pages. The second tests performs further operations on the pages in test 1: read, search for entries, and delete entries. The third test evaluates error conditions, performing illegal operations that are intended to fail. The fourth test checks boundary conditions – that is, conditions for which the page length or number of pages exceeds the maximum to test the performance of the disk space manager in those situations. Each test yields the expected results and completes successfully.

A heap files is an unordered set of records, a simple storage arrangement included within this DBMS [4]. Four tests are performed for this functionality. The first test creates a heap file, adds 100 records to the file, then scans those records. The second test opens the first file, deletes half the records, then scans the rest. The third test opens the same file, modifies those records, and ensures that the modifications are actually present. The fourth test attempts to perform illegal operations, including changing the size of a record or inserting a record that is too long, which should not be allowed. Each of these tests run as expected and are completed successfully.

The index system is used for indexing the DBMS and for running scans on the records within the system. Three tests are performed on the index system, each of which attempts to create or open a BTree index, perform an index scan, and to evaluate various possible errors that could occur on that system. Each test completes successfully.

The sorting functionality sorts the elements within the database to ensure that they are stored in the proper order within the B+ tree. Four tests are performed which attempt to sort certain elements, evaluates whether the sort has been performed successfully, and reports various error conditions. Currently, the tests evaluate integer and floating-point values, both of which are sorted successfully. All tests yield the desired result.

The SQL queries involves sending SQL commands to the DBMS for a specific set of data (sailors with boat reservations, ratings, ages, and ID numbers) to ensure that SQL commands behave properly within this Minibase DBMS. Specifically, each test evaluates the functionality of a certain number of standard SQL functionalities, including, FileScan, Sort-Merge Join, Multiple Selection, Duplication elimination, and projection. The queries themselves, along with the intended results of the queries, are included within Appendix A; the results of the queries are as intended and so each of the six query tests is completed successfully.

**Conclusion**

Each test performed within this DBMS evaluates a specific functionality of the larger database management system. Some of these tests are performed automatically, while other tests deal with user input and so they require that a user provide input to the program. For the non-interactive tests, each test runs automatically and completes successfully; this is not a surprising result as the pre-assembled DBMS should function properly before any modifications to it are made. While there is no automatic confirmation that the interactive tests ran successfully, each part of that test yields the result that would be expected for a properly running system.

Since no modifications to the original Minibase code were made, it is not surprising that the results of the tests are all successful. However, this provides insight into the functionality of each individual portion of the larger DBMS by analyzing each subcomponent. In future phases of this project, modifications to the original DBMS implementation will be made. This test functionality will evaluate the code with those alterations and will allow the user to confirm that their alterations were properly implemented to preserve the successful function of the Minibase DBMS. For now, this test code simply provides understanding of each of the individual parts of the DBMS and how they are to be run in future phases of the project.

**Bibliography**

1. C. Burch, "B+ Trees," in *Hendrix College Computer Science*, 2014. [Online]. Available: <http://1.http://www.cburch.com/cs/340/reading/btree/>. Accessed: Feb. 1, 2017.
2. O. Zaiane, "Buffer Management," in *Simon Fraser University Computer Science*, 1995. [Online]. Available: <http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chap7/node11.html>. Accessed: Feb. 2, 2017.
3. R. Ramakrishnan and J. Gehrke, "Disk Space Manager," in *University of Wisconsin-Madison Computer Science*, 1997. [Online]. Available: <http://pages.cs.wisc.edu/~dbbook/openAccess/Minibase/spaceMgr/dsm.html>. Accessed: Feb. 4, 2017.
4. R. Ramakrishnan and J. Gehrke, "Heap File," in *University of Wisconsin-Madison Computer Science*, 1997. [Online]. Available: <http://pages.cs.wisc.edu/~dbbook/openAccess/Minibase/spaceMgr/heap_file.html>. Accessed: Feb. 4, 2017.

**Appendix A: Text of Typescript Output**

**Note:** For improved readability, the MENU isnot shown for all occurrences after the first. Only the “----MENU----“ header will be included.

Script started on Wed 01 Feb 2017 01:54:06 PM MST

[01;32muser@user-Linux[01;34m ~/Documents/CSE510/minjava/javaminibase/src $[00m make test

cd tests; make bmtest dbtest; whoami; make hftest bttest indextest jointest sorttest sortmerge

make[1]: Entering directory `/home/user/Documents/CSE510/minjava/javaminibase/src/tests'

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java BMTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.BMTest

Running Buffer Management tests....

Replacer: Clock

Test 1 does a simple test of normal buffer manager operations:

- Allocate a bunch of new pages

- Write something on each one

- Read that something back from each one

(because we're buffering, this is where most of the writes happen)

- Free the pages again

Test 1 completed successfully.

Test 2 exercises some illegal buffer manager operations:

- Try to pin more pages than there are frames

\*\*\* Pinning too many pages

--> Failed as expected

- Try to free a doubly-pinned page

\*\*\* Freeing a pinned page

--> Failed as expected

- Try to unpin a page not in the buffer pool

\*\*\* Unpinning a page not in the buffer pool

--> Failed as expected

Test 2 completed successfully.

Test 3 exercises some of the internals of the buffer manager

- Allocate and dirty some new pages, one at a time, and leave some pinned

- Read the pages

Test 3 completed successfully.

...Buffer Management tests completely successfully.

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java DBTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.DBTest

Running Disk Space Management tests....

Replacer: Clock

Test 1 creates a new database and does some tests of normal operations:

- Add some file entries

- Allocate a run of pages

- Write something on some of them

- Deallocate the rest of them

Test 1 completed successfully.

Test 2 opens the database created in test 1 and does some further tests:

- Delete some of the file entries

- Look up file entries that should still be there

- Read stuff back from pages we wrote in test 1

Test 2 completed successfully.

Test 3 tests for some error conditions:

- Look up a deleted file entry

\*\*\*\* Looking up a deleted file entry

--> Failed as expected

- Try to delete a deleted entry again

\*\*\*\* Delete a deleted file entry again

--> Failed as expected

- Try to delete a nonexistent file entry

\*\*\*\* Deleting a nonexistent file entry

--> Failed as expected

- Look up a nonexistent file entry

\*\*\*\* Looking up a nonexistent file entry

--> Failed as expected

- Try to add a file entry that's already there

\*\*\*\* Adding a duplicate file entry

--> Failed as expected

- Try to add a file entry whose name is too long

\*\*\*\* Adding a file entry with too long a name

--> Failed as expected

- Try to allocate a run of pages that's too long

\*\*\*\* Allocating a run that's too long

--> Failed as expected

- Try to allocate a negative run of pages

\*\*\*\* Allocating a negative run

--> Failed as expected

- Try to deallocate a negative run of pages

\*\*\*\* Deallocating a negative run

--> Failed as expected

Test 3 completed successfully.

Test 4 tests some boundary conditions.

(These tests are very implementation-specific.)

- Make sure no pages are pinned

- Allocate all pages remaining after DB overhead is accounted for

- Attempt to allocate one more page

\*\*\*\* Allocating one additional page

--> Failed as expected

- Free some of the allocated pages

- Allocate some of the just-freed pages

- Free two continued run of the allocated pages

- Allocate back number of pages equal to the just freed pages

- Add enough file entries that the directory must surpass a page

- Make sure that the directory has taken up an extra page: try to

allocate more pages than should be available

\*\*\*\* Allocating more pages than are now available

--> Failed as expected

- At this point, all pages should be claimed. Try to allocateone more.

\*\*\*\* Allocating one more page than there is

--> Failed as expected

- Free the last two pages: this tests a boundary condition in the space map.

Test 4 completed successfully.

...Disk Space Management tests completely successfully.

make[1]: Leaving directory `/home/user/Documents/CSE510/minjava/javaminibase/src/tests'

user

make[1]: Entering directory `/home/user/Documents/CSE510/minjava/javaminibase/src/tests'

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java HFTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.HFTest

Running Heap File tests....

Replacer: Clock

Test 1: Insert and scan fixed-size records

- Create a heap file

- Add 100 records to the file

- Scan the records just inserted

Test 1 completed successfully.

Test 2: Delete fixed-size records

- Open the same heap file as test 1

- Delete half the records

- Scan the remaining records

Test 2 completed successfully.

Test 3: Update fixed-size records

- Open the same heap file as tests 1 and 2

- Change the records

- Check that the updates are really there

Test 3 completed successfully.

Test 4: Test some error conditions

- Try to change the size of a record

\*\*\*\* Shortening a record

--> Failed as expected

\*\*\*\* Lengthening a record

--> Failed as expected

- Try to insert a record that's too long

\*\*\*\* Inserting a too-long record

--> Failed as expected

Test 4 completed successfully.

...Heap File tests completely successfully.

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java BTTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.BTTest

Replacer: Clock

Running tests....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA0 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

[0] Naive delete (new file)

[1] Full delete(Default) (new file)

[2] Print the B+ Tree Structure

[3] Print All Leaf Pages

[4] Choose a Page to Print

---Integer Key (for choices [6]-[14]) ---

[5] Insert a Record

[6] Delete a Record

[7] Test1 (new file): insert n records in order

[8] Test2 (new file): insert n records in reverse order

[9] Test3 (new file): insert n records in random order

[10] Test4 (new file): insert n records in random order

and delete m records randomly

[11] Delete some records

[12] Initialize a Scan

[13] Scan the next Record

[14] Delete the just-scanned record

---String Key (for choice [15]) ---

[15] Test5 (new file): insert n records in random order

and delete m records randomly.

[16] Close the file

[17] Open which file (input an integer for the file name):

[18] Destroy which file (input an integer for the file name):

[19] Quit!

Hi, make your choice :5

Please input the integer key to insert:

4

-------------------------- MENU ------------------

Hi, make your choice :5

Please input the integer key to insert:

5

-------------------------- MENU ------------------

Hi, make your choice :5

Please input the integer key to insert:

6

-------------------------- MENU ------------------

Hi, make your choice :5

Please input the integer key to insert:

7

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 3

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 3

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

1 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

2 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

3 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :4

Please input the page number:

2

Sorry!!! This page is neither Index nor Leaf page.

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 3

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA1 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

The Tree is Empty!!!

-------------------------- MENU ------------------

Hi, make your choice :17

1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA1 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :17

0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA0 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 3

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

1 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

2 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

3 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :7

Please input the number of keys to insert:

10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA2 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 6

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (0, [ 0 0 ] )

1 (key, [pageNo, slotNo]): (1, [ 1 1 ] )

2 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

3 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

4 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

5 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

6 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

7 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

8 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

9 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 6

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 6

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice : 8

Please input the number of keys to insert:

10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA3 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 8

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 8

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (1, [ 1 1 ] )

1 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

2 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

3 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

4 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

5 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

6 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

7 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

8 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

9 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :9

Please input the number of keys to insert:

10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA4 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 10

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 10

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (0, [ 0 0 ] )

1 (key, [pageNo, slotNo]): (1, [ 1 1 ] )

2 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

3 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

4 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

5 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

6 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

7 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

8 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

9 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :10

Please input the number of keys to insert:

15

Please input the number of keys to delete:

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA5 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 12

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 12

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

1 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

2 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

3 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

4 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

5 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

6 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

7 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

8 (key, [pageNo, slotNo]): (13, [ 13 13 ] )

9 (key, [pageNo, slotNo]): (14, [ 14 14 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :6

Please input the integer key to delete:

1

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 12

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

1 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

2 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

3 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

4 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

5 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

6 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

7 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

8 (key, [pageNo, slotNo]): (13, [ 13 13 ] )

9 (key, [pageNo, slotNo]): (14, [ 14 14 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :6

Please input the integer key to delete:

2

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 12

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

1 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

2 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

3 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

4 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

5 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

6 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

7 (key, [pageNo, slotNo]): (13, [ 13 13 ] )

8 (key, [pageNo, slotNo]): (14, [ 14 14 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :11

Please input the LOWER integer key(>=0):

3

Please input the HIGHER integer key(>=0)

7

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 12

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 12

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

1 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

2 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

3 (key, [pageNo, slotNo]): (13, [ 13 13 ] )

4 (key, [pageNo, slotNo]): (14, [ 14 14 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :11

Please input the LOWER integer key(>=0):

13

Please input the HIGHER integer key(>=0)

14

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 12

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 12

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

1 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

2 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :16

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You close the file: AAA5 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :17

0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA0 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :17

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA5 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :16

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You close the file: AAA5 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :2

java.lang.NullPointerException

at btree.BT.printBTree(BT.java:344)

at tests.BTDriver.runAllTests(BTTest.java:170)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :3

java.lang.NullPointerException

at btree.BT.printAllLeafPages(BT.java:415)

at tests.BTDriver.runAllTests(BTTest.java:173)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :4

Please input the page number:

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 3

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

1 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

2 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

3 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :5

Please input the integer key to insert:

1

java.lang.NullPointerException

at btree.BTreeFile.insert(BTreeFile.java:379)

at tests.BTDriver.runAllTests(BTTest.java:188)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :2

java.lang.NullPointerException

at btree.BT.printAllLeafPages(BT.java:415)

at tests.BTDriver.runAllTests(BTTest.java:173)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :17

0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA0 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :11

Please input the LOWER integer key(>=0):

1

Please input the HIGHER integer key(>=0)

4

-------------------------- MENU ------------------

Hi, make your choice :13

java.lang.NullPointerException

at tests.BTDriver.runAllTests(BTTest.java:271)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :12

Please input the LOWER integer key (null if -3):

4

Please input the HIGHER integer key (null if -2):

8

-------------------------- MENU ------------------

Hi, make your choice :13

SCAN RESULT: 5 [ 5 5 ]

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 3

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 3

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

1 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

2 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :13

SCAN RESULT: 6 [ 6 6 ]

-------------------------- MENU ------------------

Hi, make your choice :14

-------------------------- MENU ------------------

Hi, make your choice :2

---------------The B+ Tree Structure---------------

1 3

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 3

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

1 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :7

Please input the number of keys to insert:

8900

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* The file name is: AAA6 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

---------------The B+ Tree Structure---------------

1 119

2 34

3 32

3 33

3 35

3 36

3 37

3 38

3 39

3 40

3 41

3 42

3 43

3 44

3 45

3 46

3 47

3 48

3 49

3 50

3 51

3 52

3 53

3 54

3 55

3 56

3 57

3 58

3 59

3 60

3 61

3 62

3 63

3 64

3 65

3 66

3 67

3 68

3 69

3 70

3 71

3 72

3 73

3 74

2 118

3 75

3 76

3 77

3 78

3 79

3 80

3 81

3 82

3 83

3 84

3 85

3 86

3 87

3 88

3 89

3 90

3 91

3 92

3 93

3 94

3 95

3 96

3 97

3 98

3 99

3 100

3 101

3 102

3 103

3 104

3 105

3 106

3 107

3 108

3 109

3 110

3 111

3 112

3 113

3 114

3 115

3 116

2 162

3 117

3 120

3 121

3 122

3 123

3 124

3 125

3 126

3 127

3 128

3 129

3 130

3 131

3 132

3 133

3 134

3 135

3 136

3 137

3 138

3 139

3 140

3 141

3 142

3 143

3 144

3 145

3 146

3 147

3 148

3 149

3 150

3 151

3 152

3 153

3 154

3 155

3 156

3 157

3 158

3 159

3 160

2 205

3 161

3 163

3 164

3 165

3 166

3 167

3 168

3 169

3 170

3 171

3 172

3 173

3 174

3 175

3 176

3 177

3 178

3 179

3 180

3 181

3 182

3 183

3 184

3 185

3 186

3 187

3 188

3 189

3 190

3 191

3 192

3 193

3 194

3 195

3 196

3 197

3 198

3 199

3 200

3 201

3 202

3 203

2 248

3 204

3 206

3 207

3 208

3 209

3 210

3 211

3 212

3 213

3 214

3 215

3 216

3 217

3 218

3 219

3 220

3 221

3 222

3 223

3 224

3 225

3 226

3 227

3 228

3 229

3 230

3 231

3 232

3 233

3 234

3 235

3 236

3 237

3 238

3 239

3 240

3 241

3 242

3 243

3 244

3 245

3 246

2 291

3 247

3 249

3 250

3 251

3 252

3 253

3 254

3 255

3 256

3 257

3 258

3 259

3 260

3 261

3 262

3 263

3 264

3 265

3 266

3 267

3 268

3 269

3 270

3 271

3 272

3 273

3 274

3 275

3 276

3 277

3 278

3 279

3 280

3 281

3 282

3 283

3 284

3 285

3 286

3 287

3 288

3 289

3 290

3 292

3 293

3 294

3 295

3 296

3 297

3 298

3 299

3 300

3 301

3 302

--------------- End ---------------

-------------------------- MENU ------------------

Hi, make your choice :18

0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You destroy the file: AAA0 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

java.lang.NullPointerException

at btree.BT.printAllLeafPages(BT.java:415)

at tests.BTDriver.runAllTests(BTTest.java:173)

at tests.BTDriver.runTests(BTTest.java:80)

at tests.BTTest.main(BTTest.java:648)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! Something is wrong !!

!! Is your DB full? then exit. rerun it! !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

-------------------------- MENU ------------------

Hi, make your choice :17

1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA1 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

The Tree is Empty!!!

-------------------------- MENU ------------------

Hi, make your choice :17

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* You open the file: AAA3 \*\*\*\*\*\*\*\*\*\*

-------------------------- MENU ------------------

Hi, make your choice :3

---------------The B+ Tree Leaf Pages---------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*To Print an Leaf Page \*\*\*\*\*\*\*\*

Current Page ID: 8

Left Link : -1

Right Link : -1

0 (key, [pageNo, slotNo]): (1, [ 1 1 ] )

1 (key, [pageNo, slotNo]): (2, [ 2 2 ] )

2 (key, [pageNo, slotNo]): (3, [ 3 3 ] )

3 (key, [pageNo, slotNo]): (4, [ 4 4 ] )

4 (key, [pageNo, slotNo]): (5, [ 5 5 ] )

5 (key, [pageNo, slotNo]): (6, [ 6 6 ] )

6 (key, [pageNo, slotNo]): (7, [ 7 7 ] )

7 (key, [pageNo, slotNo]): (8, [ 8 8 ] )

8 (key, [pageNo, slotNo]): (9, [ 9 9 ] )

9 (key, [pageNo, slotNo]): (10, [ 10 10 ] )

\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*

------------- All Leaf Pages Have Been Printed --------

-------------------------- MENU ------------------

Hi, make your choice :19

... Finished .

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java IndexTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.IndexTest

Running Index tests....

Replacer: Clock

------------------------ TEST 1 --------------------------

BTreeIndex created successfully.

BTreeIndex file created successfully.

Test1 -- Index Scan OK

------------------- TEST 1 completed ---------------------

------------------------ TEST 2 --------------------------

BTreeIndex opened successfully.

Test2 -- Index Scan OK

------------------- TEST 2 completed ---------------------

------------------------ TEST 3 --------------------------

BTreeIndex created successfully.

BTreeIndex file created successfully.

Test3 -- Index scan on int key OK

------------------- TEST 3 completed ---------------------

...Index tests

completely successfully

.

Index tests completed successfully

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java JoinTest.java

Note: JoinTest.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.JoinTest

Replacer: Clock

Any resemblance of persons in this database to people living or dead

is purely coincidental. The contents of this database do not reflect

the views of the University, the Computer Sciences Department or the

developers...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query1 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved boat number 1.

and print out the date of reservation.

SELECT S.sname, R.date

FROM Sailors S, Reserves R

WHERE S.sid = R.sid AND R.bid = 1

(Tests FileScan, Projection, and Sort-Merge Join)

[Mike Carey, 05/10/95]

[David Dewitt, 05/11/95]

[Jeff Naughton, 05/12/95]

Query1 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query1 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query2 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved a red boat

and return them in alphabetical order.

SELECT S.sname

FROM Sailors S, Boats B, Reserves R

WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'

ORDER BY S.sname

Plan used:

Sort (Pi(sname) (Sigma(B.color='red') |><| Pi(sname, bid) (S |><| R)))

(Tests File scan, Index scan ,Projection, index selection,

sort and simple nested-loop join.)

After Building btree index on sailors.sid.

[David Dewitt]

[Mike Carey]

[Raghu Ramakrishnan]

[Yannis Ioannidis]

Query2 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query2 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query3 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved a boat.

SELECT S.sname

FROM Sailors S, Reserves R

WHERE S.sid = R.sid

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]

[Mike Carey]

[Mike Carey]

[David Dewitt]

[David Dewitt]

[Jeff Naughton]

[Miron Livny]

[Yannis Ioannidis]

[Raghu Ramakrishnan]

[Raghu Ramakrishnan]

Query3 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query3 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query4 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved a boat

and print each name once.

SELECT DISTINCT S.sname

FROM Sailors S, Reserves R

WHERE S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]

[Jeff Naughton]

[Mike Carey]

[Miron Livny]

[Raghu Ramakrishnan]

[Yannis Ioannidis]

Query4 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query4 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query5 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of old sailors or sailors with a rating less

than 7, who have reserved a boat, (perhaps to increase the

amount they have to pay to make a reservation).

SELECT S.sname, S.rating, S.age

FROM Sailors S, Reserves R

WHERE S.sid = R.sid and (S.age > 40 || S.rating < 7)

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]

[Mike Carey, 9, 40.3]

[Mike Carey, 9, 40.3]

[David Dewitt, 10, 47.2]

[David Dewitt, 10, 47.2]

[Jeff Naughton, 5, 35.0]

[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query5 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query6 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors with a rating greater than 7

who have reserved a red boat, and print them out in sorted order.

SELECT S.sname

FROM Sailors S, Boats B, Reserves R

WHERE S.sid = R.sid AND S.rating > 7 AND R.bid = B.bid

AND B.color = 'red'

ORDER BY S.name

Plan used:

Sort(Pi(sname) (Sigma(B.color='red') |><| Pi(sname, bid) (Sigma(S.rating > 7) |><| R)))

(Tests FileScan, Multiple Selection, Projection,sort and nested-loop join.)

After nested loop join S.sid|><|R.sid.

After nested loop join R.bid|><|B.bid AND B.color=red.

After sorting the output tuples.

[David Dewitt]

[Mike Carey]

[Raghu Ramakrishnan]

[Yannis Ioannidis]

Query6 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query6 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Finished joins testing

join tests completed successfully

/usr/lib/jvm/default-java/bin/javac -classpath .:.. TestDriver.java SortTest.java

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.SortTest

Running Sort tests....

Replacer: Clock

------------------------ TEST 1 --------------------------

Test1 -- Sorting OK

------------------- TEST 1 completed ---------------------

------------------------ TEST 2 --------------------------

Test2 -- Sorting OK

------------------- TEST 2 completed ---------------------

------------------------ TEST 3 --------------------------

-- Sorting in ascending order on the int field --

Test3 -- Sorting of int field OK

-- Sorting in descending order on the float field --

Test3 -- Sorting of float field OK

------------------- TEST 3 completed ---------------------

------------------------ TEST 4 --------------------------

Test4 -- Sorting OK

------------------- TEST 4 completed ---------------------

...Sort tests

completely successfully

.

Sorting tests completed successfully

/usr/lib/jvm/default-java/bin/javac -classpath .:.. SM\_JoinTest.java TestDriver.java

Note: SM\_JoinTest.java uses unchecked or unsafe operations.

Note: Recompile with -Xlint:unchecked for details.

/usr/lib/jvm/default-java/bin/java -classpath .:.. tests.SM\_JoinTest

Replacer: Clock

Any resemblance of persons in this database to people living or dead

is purely coincidental. The contents of this database do not reflect

the views of the University, the Computer Sciences Department or the

developers...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query1 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved boat number 1.

and print out the date of reservation.

SELECT S.sname, R.date

FROM Sailors S, Reserves R

WHERE S.sid = R.sid AND R.bid = 1

(Tests FileScan, Projection, and Sort-Merge Join)

[Mike Carey, 05/10/95]

[David Dewitt, 05/11/95]

[Jeff Naughton, 05/12/95]

Query1 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query1 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query3 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved a boat.

SELECT S.sname

FROM Sailors S, Reserves R

WHERE S.sid = R.sid

(Tests FileScan, Projection, and SortMerge Join.)

[Mike Carey]

[Mike Carey]

[Mike Carey]

[David Dewitt]

[David Dewitt]

[Jeff Naughton]

[Miron Livny]

[Yannis Ioannidis]

[Raghu Ramakrishnan]

[Raghu Ramakrishnan]

Query3 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query3 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query4 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of sailors who have reserved a boat

and print each name once.

SELECT DISTINCT S.sname

FROM Sailors S, Reserves R

WHERE S.sid = R.sid

(Tests FileScan, Projection, Sort-Merge Join and Duplication elimination.)

[David Dewitt]

[Jeff Naughton]

[Mike Carey]

[Miron Livny]

[Raghu Ramakrishnan]

[Yannis Ioannidis]

Query4 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query4 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query5 strating \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query: Find the names of old sailors or sailors with a rating less

than 7, who have reserved a boat, (perhaps to increase the

amount they have to pay to make a reservation).

SELECT S.sname, S.rating, S.age

FROM Sailors S, Reserves R

WHERE S.sid = R.sid and (S.age > 40 || S.rating < 7)

(Tests FileScan, Multiple Selection, Projection, and Sort-Merge Join.)

[Mike Carey, 9, 40.3]

[Mike Carey, 9, 40.3]

[Mike Carey, 9, 40.3]

[David Dewitt, 10, 47.2]

[David Dewitt, 10, 47.2]

[Jeff Naughton, 5, 35.0]

[Yannis Ioannidis, 8, 40.2]

Query5 completed successfully!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Query5 finished!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Finished joins testing

join tests completed successfully

make[1]: Leaving directory `/home/user/Documents/CSE510/minjava/javaminibase/src/tests'

[01;32muser@user-Linux[01;34m ~/Documents/CSE510/minjava/javaminibase/src $[00m exit

Script done on Wed 01 Feb 2017 02:03:14 PM MST