CSE 510 Phase 1

Database Management Systems Implementation

Mishal Shah
1217195421
mshah31@asu.edu

Samip Thakkar
1217104967
sthakka2@asu.edu

Saurabh Gaggar
1219666643
sgaggar1@asu.edu

Khushal Modi
1219446332
kcmodi@asu.edu

Kunj Patel
1213184152
khpatel8@asu.edu

Monil Nisar
1217111805
mnisar2@asu.edu

February 1, 2021

1 Abstract

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS has functions for modification, creation and deletion for data and its definitions. The administrative aspect of DBMS takes care of data security, monitoring users, performance, data integrity maintenance, concurrency control and information recovery from unexpected system failure.

2 Introduction

Minibase comprises some components which perform specific operations. These components are as follows:

2.1 Buffer Manager

The goal of a database management system is to minimise the number of transfers between disk and memory. This can be done by keeping as many blocks of data in main memory which reduces accesses to disk. Database buffer is the temporary storage for storing copy of disk blocks. A Buffer manager is responsible for allocation of space to store data in the buffer. It implements a replacement policy to decide which block to evict when the buffer is full.

2.2 Disk Manager

Disk manager manages space on the disk. It keeps track of which disk blocks are in usage. It hides the details of underlying hardware and OS to allow higher levels of software to think data as collection of blocks. It can also help distribute data to multiple disks.

2.3 Heap Files

Simplest form of file organisations. Records are inserted at the end of the file as and when inserted with no ordering or sorting. If the data block is full, new record can be written in any of the other blocks. Blocks are not selected sequentially for writing. Works for bulk data insertions but search and modification operations are costly.

2.4 B Trees

B-trees store data in nodes in sorted manner. It stores data such that each node contains key in ascending order. Each of the keys has reference to other two children nodes. The left child node keys are less than the current key and right side child node keys are more than the current key. It allows searches, sequential access, insertions and deletions in logarithmic time. Well suited for storage systems to read/write large blocks of data.

2.5 Index

Indexing in DBMS is used to optimise the performance of the database by reducing the number of disk accesses required when query is processed. Can be used to locate and access data in databases quickly.

2.6 Join

A Cartesian product of two relations gives us all possible tuples that can be paired together. It might not work in certain cases to take a cartesian product when we have huge relations. Join combines Cartesian product followed by a selection process. Join operation pairs two tuples from different relations only when join condition is satisfied.

2.7 SortMerge Join

In this join, the database sorts the first row source by its join column then sorts the second row source by its join columns and merges the sorted row sources together. As matches are found, they are put in the result set. Also both sides of the join are sorted by the join predicates.

3 Description of tests

3.1 Buffer Manager tests

- The first test allocates some new pages and performs write operation on each page and subsequently reads them and finally empties the pages. This test passes successfully as every operation is legal and pages are read only after they are written.

- The second test tries to perform illegal operations. First operation pins more pages than the number of available frames. This operation fails as expected. The second operation tries to free a pinned page and the third operation tries to unpin an already unpinned page. The second and third operation fails as well.
- The third test allocates new pages, marks them ”dirty” and randomly leaves some unpinned. All newly allocated pages are pinned again and so some of pages are doubly pinned. Data written on them is checked and all pages are then unpinned. This test passes if there are no errors.

3.2 Disk Space Manager tests

- The first test creates the database and does normal operations like adding file entries and allocation of pages. Some pages will be written and rest are deallocated.
- The second test opens the database created in the first test and deletes some of the file entries. Remaining file entries are looked up. Finally data is read from pages written in test 1 and checked for correctness.
- The third test looks for error conditions such as lookup for an already deleted file entry, deleting a deleted entry/non-existent file, adding a file entry that already exists or whose name is too long, allocating a page that’s too long and allocating/deallocating negative run of pages. The operations in this test will fail as expected.
- The fourth test checks some boundary conditions. Database created in test1 is removed and a new database is created and all pages are allocated. Some of these pages are deallocated and again some are allocated. The directory size is increased to more than one page by adding file entries. Finally the database is destroyed.

3.3 Heap file tests

- Heap file is created in the first test. N fixed size records are inserted and the heap file is scanned. Records are scanned in the order in which they were inserted.
- The second test removes some of the records that were earlier inserted and the remaining records are scanned.
- The third test relates to updating of the records. Updates are performed on one of the fields in the records and finally the records are scanned to check for correctness.
- The fourth test creates a non-persistent heap file. The records are not saved to the disk. The test will check for error conditions such as changing the size of records and inserting records that are longer than allowed. The operations in this test will fail as expected.

3.4 B Tree tests

- The first test creates an index and inserts key/rid pairs in it. The keys are inserted in ascending order. The user can perform a scan on the database to print tree structure and leaf pages once all the records are inserted.

- The second test inserts the user records in reverse order. The user can print the tree structure and leaf pages once the records are inserted.
- The third test inserts the records in a random fashion. User can scan the database once the records are inserted and print the tree structure as well as the leaf pages.
- The fourth test randomly inserts records and then deletes some records in random order. The user can then scan the database to print the tree structure and the leaf pages.
- The fifth test is mostly similar to the fourth but the keys for this test will not be integer. The user can scan the database and print the tree structure and leaf pages.

3.5 Index tests

- The first test creates a B-tree Index file, inserts some records and scans the file. The test is successful if no logical errors are involved.
- The second uses the same file created in the first test and performs two operations: identity selection and range scan over the records. The test is successful if no logical errors are involved.
- The last test will create a new file and perform a scan operation after inserting some unordered records. If the index scan does not return the expected ordered values, then the test will fail. This test is expected to be successful if there are no errors with the above operations.

3.6 Join and Sort-merge tests

- The database is populated with 3 tables: Sailors, Boats, and Reserves. The first couple of tests perform a join operation with given condition between tables Sailors and reserves to find names and dates of sailors who reserved a boat. The subsequent tests do the same but with the added sort key. A number of query operations in both these tests which are as follows:
  - File scan, projection, sort-merge join
  - File scan, index scan, projection, index selection, sort, nested-loop join
  - File scan, projection, sort-merge join
  - File scan, projection, sort-merge join, duplicate elimination
  - File scan, selection, projection, sort-merge join
  - File scan, selection, projection, sort, nested-loop join

3.7 Sort tests

- The first test adds records to the database and tries to sort. After which it'll check for unsorted records or difference in the number of records. The test is successful and returns expected results.
- The second and the third test sort the index in ascending/descending order and then checks for any anomalies. The test is successful and returns expected results.
- The fourth test adds records to the database and sorts them and checks for unsorted records or different number of records. The test is successful and returns expected results.

4 Conclusion

With dive deep into the mini base components through low-level tests, we learnt how these components function to support standard database features. These tests checked the structure and operations of the database. The first few tests focused on data transfer between storage and database. The last few tests focused on standard query operations like select, join, projection and sort. Some tests also dealt with B-tree indexes to illustrate how data is stored in the database.

References

- Minibase documentation: `https://research.cs.wisc.edu/coral/minibase/minibase.html`
- Buffer manager: `https://web.stanford.edu/class/cs346/2015/notes/Lecture_One.pdf`
- Disk Space manager: `http://www.dbmsinternals.com/database-fundamentals/data-storage/disk-space-management-pages/`
- B-tree Indexes: `https://www.geeksforgeeks.org/introduction-of-b-tree/`
- Sort merge join: `https://use-the-index-luke.com/sql/join/sort-merge-join`

5 Appendix

Please find typescript file attached in the zip file.