CS 2550 – PRINCIPLES OF DATABASE SYSTEMS (SPRING 2020)
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF PITTSBURGH

Term Project: *myPTA* (Phase 1)

Release: Mar. 1, 2020 Team Declaration Due: 8PM, Thu., Mar. 5, 2020.
Due: 8PM, Thu., Mar. 26, 2020.

Project Teams

This is *team* project. Each project team will have 4 members and is composed of two homework teams. Please find your partner homework team and send an email to cs2550-staff@cs.pitt.edu, CCing all team members, to declare your project team. The deadline for team declaration is **8PM, Thursday, Mar. 5, 2020**. We will randomly assign project teams to students who have not declared theirs after the deadline.

Goal

The objective of the project is to develop *myPTA* (my Pitt Transaction Agent), a transactional row store database that efficiently supports concurrent execution of OLTP (i.e., transactions) workloads. *myPTA* will provide limited *transactional* support. By limited support, we mean that it does not support full durability. In addition to serializable and atomic access, it will also provide the standard uncontrolled access to files.

Description

myPTA consists of three components: *Data Manager, Scheduler,* and *Transaction Manager*. The *Data Manager* is responsible for managing the data on the primary and secondary storage. The *Scheduler* is responsible for scheduling operations from transactions. The *Transaction Manager* is responsible for reading in transaction operations. In the first phase (i.e., this assignment), you are required to implement the *Data Manager*.

Data Manager

For this project, all records (tuples) have the following fields (schema):

- ID: 4-byte integer (Primary Key)
- ClientName: 16-byte long string
- Phone: 12-byte long string

You are responsible to implement the following operations for the first phase:

- `R table val`: Retrieve the record with ID=val in table. If table does not exist, the read is aborted.
- `M table val`: Retrieve the record(s) which have val as area code in Phone attribute in table. If table does not exist, the read is aborted.
- `W table (t)`: Write (i.e., Insert or Update) a record t into table. If table does not exist, it is created.
- `D table`: Delete table.

All operations should be read from a script file. Below is an example of a script file (assuming we have tables X and Y):

```
----------------------------------------------
     R X 13
     R Y 7
     W Y (5, John, 412-111-2222)
     M Y 609
     W X (2, Thalia, 412-656-2212)
     M X 412
     D Y
----------------------------------------------
```

Disk Organization

You need to implement two file organization strategies, and compare their read and write performance.

The first file organization strategy is the Sequential File strategy, which is the baseline strategy. In this strategy, records are kept on persistent storage (disk) in *sequential files* in a *row store* fashion, meaning that each table is kept in a separate file. Each file consists of *slotted pages*. Each slotted page is of the size of a disk block. The size of a disk block is tunable at the beginning of the program execution.

The second file organization is the LSM-tree strategy. In this strategy, records are kept on persistent storage (disk) in a *row store* fashion in *SSTables*. SSTables have a fixed size limit of disk blocks (tunable at the beginning of the program execution). The leveled compaction strategy is to be used. There should be at most 4 SSTables in level-0, and at most 10 SSTables in level-1. Each level above level-1 can have ten times as many SSTables as the previous level. For example, level-2 can have at most 100 SSTables. SSTables in each level are compacted into the next level when the level is filled up. SSTables in each non-0 level should have non-overlapping key ranges.

Main Memory Organization

The database buffer in the main memory has a fixed size, and is set at the beginning of the program execution. The database buffer is used differently for the two file organization strategies. For the Sequential Files strategy, the buffer is used as the read/write cache. For the LSM-tree strategy, the buffer is used for the memtable, the compaction, and the read cache.

For the Sequential Files strategy, the records will have to be brought to the database buffer to be accessed. However, the number of buffer pages available in database buffer is limited. Therefore, you are required to implement the Least Recently Used (LRU) page replacement mechanism to swap pages in and out.

For the LSM-tree strategy, writes and deletes (i.e., operation W and D) are added to the *memtable*. The memtable has a fixed size twice the size limit of the SSTables. The memtable needs to be flushed at the moment when the records it contains can fill up a full level-0 SSTable.

For the LSM-tree strategy, reads (i.e., operation R and M) need to bring SSTable blocks from disk to database buffer, and the blocks will be kept in a read cache. Similar to the Sequential Files strategy, you are required to implement the Least Recently Used (LRU) for the replacement mechanism to add and remove blocks. Additionally, you need to come up with methods (e.g., synchronization or invalidation protocols) to keep the cache consistent.

All meta-data and control structures in the main memory (such as page tables) that are needed for efficient processing are kept outside of the database buffer. As opposed to the database buffer, there is always sufficient space for meta-data and control structures.

Logging

Data Manager should keep a log file in which it can record all its actions. The logging formats for the two file organizations are slightly different, and we will introduce them one by one below.

For the Sequential Files strategy, you need to record all Data Manager actions such as performed operations, creating new pages and swapping existing pages in and out of main memory. Following is the accepted logging format:

```
----------------------------------------------
     W X (1, Alex, 412-111-1111)
     CREATE T-X P-1
     SWAP IN T-X P-1
     Written: X, 1, Alex, 412-111-1111
     W X (2, Bob, 412-222-2222)
     Written: X, 2, Bob, 412-222-2222
     ...
     SWAP OUT T-X P-1
     ...
     R X 1
     SWAP IN T-X P-1
     Read: X, 1, Alex, 412-111-1111
     D Y
     Deleted: Y
     M X 412
     MRead: X, 1, Alex, 412-111-1111
     MRead: X, 2, Bob, 412-222-2222
----------------------------------------------
```

Note that `T-X P-1` means Table X, Page 1.

For the LSM-tree strategy, you need to record all Data Manager actions such as performed operations, creating new SSTables and swapping existing SSTable blocks into the database buffer. Following is the accepted logging format:

```
----------------------------------------------
     W X (1, Alex, 412-111-1111)
     Written: X, 1, Alex, 412-111-1111
     W X (2, Bob, 412-222-2222)
     Written: X, 2, Bob, 412-222-2222
     CREATE L-0 K-X1-X2
     R X 1
     SWAP IN L-0 K-X1-X2
     Read: X, 1, Alex, 412-111-1111
     D Y
     Deleted: Y
     M X 412
     MRead: X, 1, Alex, 412-111-1111
     MRead: X, 2, Bob, 412-222-2222
----------------------------------------------
```

Note that `CREATE` means flushing a new SSTable to the disk either from the memtable or from the compaction process. `SWAP IN` means swapping in a SSTable for compaction or a SSTable block for reading. `L-0 K-X1-X2` means Level 0, Key range from table X key 1 to table X key 2 inclusive.

Logging can significantly help you with debugging your project. In following phases, these log records will be modified and utilized to restore the database to the previous consistent state if a transaction is aborted.

In the following phases of this project your job will be to keep the data synchronized and consistent between the memory, and the disk while executing multiple transactions simultaneously. In this phase you just need to make sure that a single script file can execute from the beginning to the end producing correct output (log) and leaving the database in a consistent state.

Documentation

You need to write a detailed documentation about your design and implementation of the Data Manager for *myPTA*. The documentation should cover at least the following aspects:

1. For the Sequential Files strategy:
    - Description about the record placement, file organization, and memory management strategies.
    - Detailed description of the implementation of the slotted page (e.g., how is the record order maintained in the page and among pages).
    - Test cases indicating the correctness of the four operations under different conditions.
2. For the LSM-tree strategy:
    - Description about the record placement, file organization, and memory management strategies.
    - The design and components of the row key (i.e. the key for each record).
    - The design of your read cache, and the methods to keep it consistent.
    - For operation W: Describe all processes that could be triggered by an operation W. Detailed description of the compaction strategies (e.g., how many and which SSTable(s) to pick for compaction in each level?).
    - For operation D: Describe all processes that could be triggered by an operation D.
    - For operation R: How to guarantee the correctness of the read under different conditions (e.g., multiple updates, deleted tables).
    - For operation M: Have you implemented any methods other than the full table scan to do it?
    - Any other implemented optimization methods, e.g., additional index files, auxiliary memory objects, parallel compaction, etc.
    - Test cases indicating the correctness of the four operations under different conditions.
3. For the comparison between the two file organization strategies:
    - The comparison and analysis of the read/write throughput (records per second) between the two strategies under different combinations of buffer sizes and SSTable sizes.
4. For any implemented optimization methods of the LSM-tree strategy:
    - The comparison and analysis of the read/write throughput (records per second) between the LSM-tree strategy without the optimization and with the optimization.
5. Others
    - Detailed description of the contribution of each team member based on the implemented components (e.g., for the LSM-tree strategy, who implemented the memtable, the SSTable, the compaction algorithms?).

Submission Guidelines

- The project code will be collected by the TA via the GitHub repository that you share with only your TA (xzhangedu) and your team members. To turn in your code, you must do three things by the deadline:

1. Make a commit to your project repository that represents what should be graded as your group's submission for this phase. The message for this commit should be "Phase 1 submission".
2. Push that commit to the GitHub repository that you have shared with the TA. Please make sure that the commit is pushed before the due date.
3. Prepare a link to your GitHub repository, the commit ID (Hash) of the commit to be graded and a suitable title for the submission - i.e. Project Phase 1 Submission. Send this information into an email to cs2550-staff@cs.pitt.edu. Submit the same information into a document, named "Project Phase 1 Submission" via the assignment submission web interface. Commit ID can be found on GitHub or as part of output of "gitlog" command.

- The following items are required to be in the GitHub repository to be collected:
  - Your source code (You have the option to implement your prototype *myPTA* in any language and on Windows or Unix-based operating system).
  - The documentation file.
  - Your test input script files for all documented experiments.
  - A readme file about how to compile the code, run the program, and where to find the output log files.
- You may submit Phase 1 multiple times. The last commit before the due date (**8PM, Thursday, Mar. 26, 2020**) will be graded.

Grading

The grading of the project phase 1 is based on the following aspects:

- The successful compilation and running of the source code.
- The quality of the documentation.
- The comprehensiveness and completeness of the test cases.
- Implemented optimization methods for the LSM-tree strategy.

Academic Honesty

The work in this project is to be done *independently* by each group. Discussions with other groups on the assignment should be limited to understanding the statement of the problem. Cheating in any way, including giving your work to someone else will result in an F for the course and a report to the appropriate University authority.