VE482 Project 2

LemonDB

Work Document

Team 3:

Wang Ren

Wu Wentao

Zhang Jiaruo

Liu Boning

Contents

1 Introduction

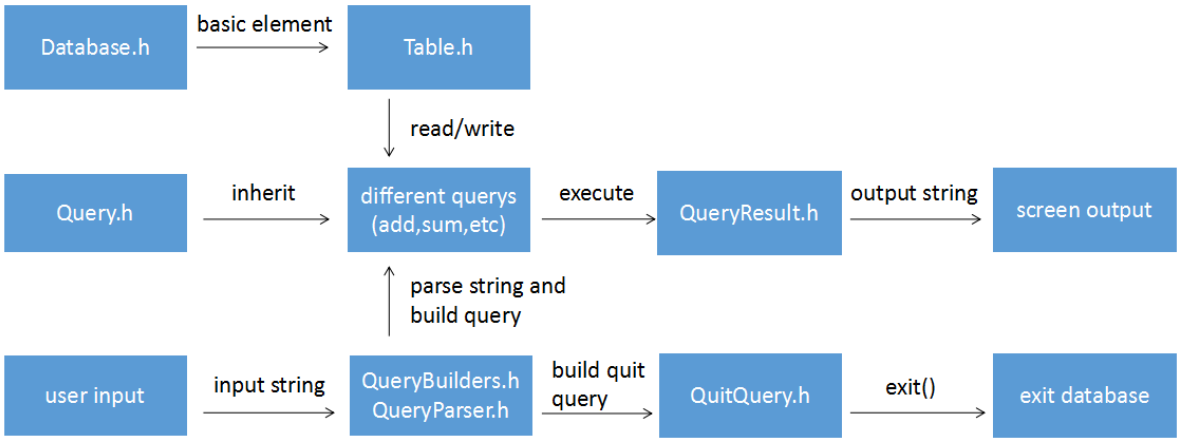
1.1 Purpose of this document

The main purpose of this document is to let users and developers get familiar with lemon database (team3 version). In this document, I will describe how the database works and explain its algorithm. Besides, I will also provide some tips for users and developers.

1.2 Feedback and correction

If you have any questions or find any mistakes about the document, please contact our team by email: liuboning98@sjtu.edu.cn, we will give you feedback as soon as we can.

2 Structure of database



2.1 Database

The Database class is declared in Database.h, which is the main structure containing all the data information.

2.2 Table

The Table class is declared in Table.h. In our database, the basic element

is table, which contains all the data and information we need to read or write. A table consists of keys, fields and corresponding data.

2.3 Query

The Query class is declared in Query.h. Query is the type of command, and with each query users can perform a specific read/write command to operate information from tables. In order to do different queries, we build different subclasses inherited from Query class, such as AddQuery, SumQuery, etc. Each query class is respectively declared in a head file.

2.4 Input process

The user input is string command, so we use QueryBuilders and QueryParsers, which are declared in QueryBuilders.h and QueryParsers.h. QueryBuilders contain methods to get the required part of a query from the input string and QueryParsers use the methods to parse a string input and return the corresponding query object, which is executable.

2.5 Execution and result

The execution function of each kind of query is realized in each query's cpp file. During execution, some write commands like AddQuery will change the corresponding table object, and read command like sum will not change table object. After execution, a QueryResult object, which is declared in QueryResult.h, will be returned, and we use it later to output information we get using a certain query.

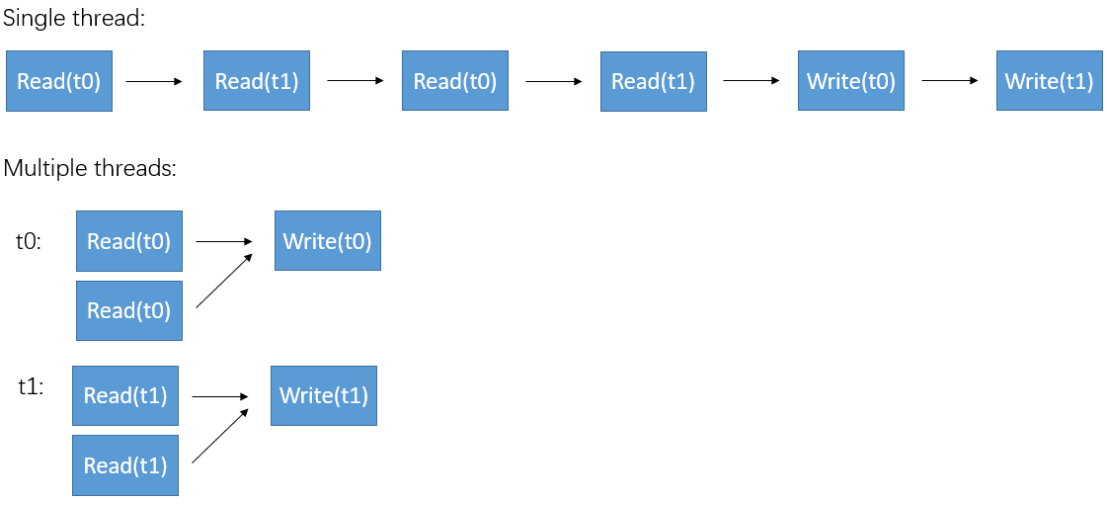
2.6 Exit

Specially, if the parsed and built query is QuitQuery, then the member function exit() of the database object will be called, and user exit the database.

3 Algorithm of multiple threads

3.1 Improvement of performance

The basic implement of database allow us to execute queries one by one. However, when getting large amounts of queries, it's necessary to execute them synchronously. The improvement of performance is obvious, since we can read information from a table at the same time, or operate on different tables synchronously.



3.2 LISTEN

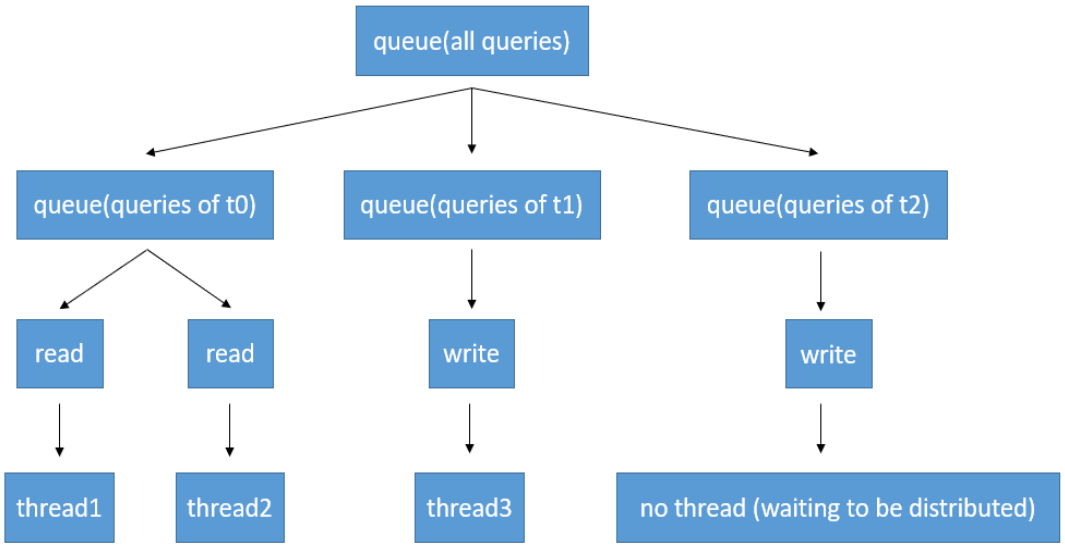
LISTEN reads queries from a file one by one and puts them into a queue, so that these queries can be taken out in order.

3.3 Scheduling in execution

Scheduling block takes queries from the query queue and puts them into different queues representing corresponding tables (one queue for one table). Threads are distributed to these queues to implement synchronous execution.

Every table (corresponding to its queue) is a reader-writer problem. It needs a read lock and write lock to prevent race condition.

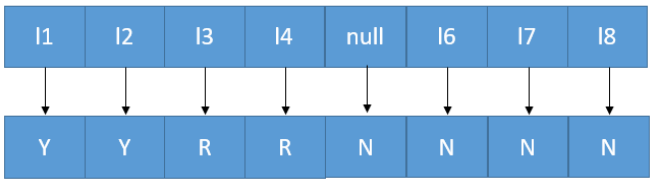
Scheduling thread will check all the table queries one by one circularly, if the top query in a queue is unlocked, then scheduling thread will distribute a free thread to it if there're threads remaining, and look for the next query in the queue, else, it will look for the next queue. If the scheduling thread has gone through all the queues once and finds no queries unlocked, it will sleep for a while until there's a thread completing its job and returns an awaking signal.



A snapshot of current program (3 threads available)

3.4 Output

Output thread will grab results from all the threads after they execute the queries, and puts them into correct place corresponding to lines of queries. If all the queries whose line is smaller than a query have been output, this query should also be output.



Y: the result has been output
N: the result hasn't been output
R: the result is ready to be output

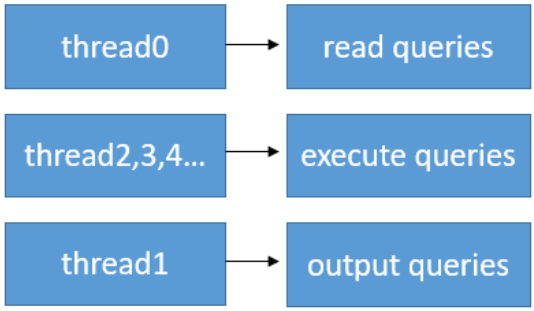
3.5 Problems overcome and overview

The main problem of multiple thread is the critical region of the database. The critical region should be separated into several parts, representing corresponding tables. Every table is a single reader-writer problem, and can be solved by adding write lock and read lock.

Another problem is the distribution of threads. Since tables are equivalent to the scheduling thread, every table should have equal chance to obtain a thread if its top query is unlocked. Thus, the scheduling thread checks all the table circularly and distributes threads to those tables within unlocked queries. If there's no unlocked query for all tables at a moment, scheduling thread will sleep for a while in order to prevent meaningless time-cost loop until a working thread awakes it or it reaches the maximum sleeping time.

Overview: the design of multiple-thread database is reading (single thread), executing (multiple thread), output (single thread). These three blocks work

at the same time to improve the performance of the database.



4. Reminders for users and developers

- The current version does not support any invalid inputs.
- This database supports standard SQL query.
- There should be no connection among tables.
- The manual of the database can be gotten in p2.pdf.