Introduction to Database Systems

Individual Homework 3: Query Optimization & Simple 2PL

1. Introduction

In the lecture, you had learned some query processing methods and how to perform concurrency control. In this homework, you are required to implement a concurrency control based on 2PL and answer some questions and observations about both query optimization and concurrency control. Please continue reading for details.

2. Tasks

2-1. Query Optimization

You have learned in class about different query strategies cause different efficiency. Also, you may find that some queries in Individual Homework 1 take much time to finish. In this part, you are going to analyze and write your observation in details. You need to read parts of the MySQL Documentation and get familiar with the MySQL query optimizer and the visualization command including EXPLAIN, SHOW PROFILE, ANALYZE, ...etc. Here we will use same datasets as Individual Homework 1.

2-1-1.

Choose the most time you spent of your sql in Individual Homework 1 and analyze your with TA's sql by using above commands of MySQL. You can choose either one of TA's sql with yours or you can compare two TAs' results from github directly. You are encouraged to use yours to compare with one of TA's to observe how different about query strategies. TA's sql would not always be efficient actually :). Please give detailed explanations about how you analyze and what's your observation. You can also use other commands not mentioned above to explain your opinion.

2-1-2.

Use **trace optimizer** in MySQL (document) to trace both yours and one of TA's task at C-6 in Individual Homework 1. Try to interpret and explain in details how efficient of two queries and what you observe. Note that you need to turn on optimizer_trace in MySQL first. Also notice that you should pick different TA than 2-1-1 you selected.

2-2. Simple 2PL

In this part, you are required to write a program to do addition and subtraction operation by using 2PL.

Description

- Fristly, you will get many variables, like $0, $1, $2… and their initial values.
- And you will get some equations like '$0 = $0 + 17 + 4', '$43 = $12 - $39 + 8 + 12', then you should perform the same operation to your variable value.
- After finish all the equations, your should write your result to a file.
- Following is the input format
    - In the first line, you will get a number ***N***, indicates that there are ***N*** variables in this task.
    - Then you will get ***N*** numbers separated by blank space, indicate the initial value for each variables ($0, $1, $2 … $N-1 respectively).
    - Following there will be many lines, indicate the operation, you should read and perform it until the end of the file.
    - Notice that the answer would be doing the operation in sequence.
    - All the values would in the range of int type.
- Example
    - Input:

example

```
3
1 4 18
$0 = $0 + 7 - 2
$1 = $1 + 6
$2 = $2 - 8 + $0 - 4
$0 = $1 + $2 + $0
```

    - Output:

example_output

```
$0 = 28
$1 = 10
$2 = 12
```

- We provide three types of data
    1. 10 variables, 100 equations. 1 to 10 operation for each task

    - The first term on the right of the equal sign will always same as the variable on the left of the equal sign. (like line 3 and line 4 of the above example)
    - The rest of the value on the right of the equal sign will always be a positive integer.
2. 70 variables, 100 equations. 1 to 10 operation for each task
    - The first term on the right of the equal sign could be different with the variable on the left of the equal sign. (like line 6 of the above example)
    - The equation would consist more than one variable. (like line 5 and line 6 of the above example)
3. 70 variables, 100 equations. 100000 to 150000 operation for each task
    - Equation format is same as data 1.

Program

- In this homework, you're required to write a **multithreaded program with 2PL** to deal with the data.
- The **implementation detail is not limit**. But as we mentioned above, you should write a multithreaded program with 2PL to perform concurrency control. We will provide suggest algorithm in the next section.
- Your are not required to write deadlock prevention, but in TA's program, deadlock is never happened. So please try to make it work fine.
- You should write your program in **C/C++**
- Please read the data from **standard input**, and redirect the data to your program.
- The **first argument** is how many thread your program should open. The **second argument** is the file name your output should write to. And you can use `diff` command to see the difference between your answer and ground truth. (You can ignore the standard output of the following example.)

```
karljackab@karl:~/DBMS20_hw3$ ./main 2 example_output < example
Thread number: 2
Time: 0.000929172 s
karljackab@karl:~/DBMS20_hw3$ cat example_output
$0 = 28
$1 = 10
$2 = 12
karljackab@karl:~/DBMS20_hw3$ diff example_answer example_output
```

- Please **write your own Makefile**, and named the output as "**main**". TA will compile your code by your makefile. (So that we just need to type `make` in the terminal, then it will compile your code to one "main" file)

Suggested Algorithm

- Please google the keywords first: “pthread”, “mutex”, “semaphore”
- You can use pthread to create thread, mutex to perform 2PL and job access, semaphore to trigger incoming jobs and finished jobs count.
- Following is the suggested algorithm
    1. Read the required thread number from argv, and create the threads. Make these thread wait the “job semaphore”.
    2. Read the variable number and initial values.
    3. Read and parse the equation, pack this equation to a single job (defined by yourself, you can use struct or class to pack them).
       Put this job into a job list (also defined by yourself, you can use vector or other data type to maintain it).
       Then trigger a “job semaphore”.
    4. One of your thread will get the job semaphore.
       After get the semaphore, access the job list and perform it. You will implement 2PL in this step.
       After finishing the job, trigger a “finished semaphore”, and loop it again to wait next job semaphore.
    5. At the end of your main program, after parsing the input data, wait the finished semaphore many times, so that we could know whether all of these jobs have finished or not.
    6. Write the result variable values to the output file.

2-2-1.

Please explain your **whole program** with your code. **Paste the screenshot of your code**, explain the working flow and what is it doing as detail as possible. You **must** point out your “growing phase” and “shrinking phase” in your code.

2-2-2.

Compare the execution time of your program in **three** types of the data we provided and **different thread number**. You can also write more test cases in your experiment and report.

2-2-3.

Does your program work fine at data 2 no matter how much the thread it use? If no, why? And how to solve that problem? If yes, paste the screenshot to show that.

3. Grading

Noted that we will highly judge your grade based on your report. Try to **make your report clear** and **as detail as possible**.

For automatic checking, **TA will use your makefile to compile your code**, and run your program in other data automatically to check the answer of your result. The data type will be same as the description we mentioned above, but the content will be different.

Plagiarism is not allowed! You will get **huge penalty** if we find that.

Description	Score(%)
Answer for 2-1-1	15
Answer for 2-1-2	20
Answer for 2-2-1	15
Answer for 2-2-2	20
Answer for 2-2-3	5
Automatic Checking for data 1	5
Automatic Checking for data 2	10
Automatic Checking for data 3	10

4. Discussion

TAs had opened a channel **HW3 討論區** on Microsoft Teams of the course, you can ask questions about the homework in the channel. TAs will answer questions in the channel as soon as possible.

Discussion rules:

1. Do not ask for answer of the homework.
2. Check if someone had asked the question you have before asking.
3. We encourage you to answer other students' questions, but again, do not give the answer of the homework. Reply the messages to answer questions.
4. Since we have this discussion channel, do not send email to ask questions about the homework unless the questions are personal and you do not want to ask publicly.

5. Submission

1. The deadline of this homework is **06/10 (Wed.) 23:59:59**, no submission allowed after **06/17 (Wed.) 23:59:59**.

2. You should put your “pdf”, “makefile” and your code into one folder, your report should named as “**HW3_XXXXXXX.pdf**”, the folder should named as “**HW3_XXXXXXX**”, where XXXXXXX is your student ID.

   Please **make your Makefile work fine**!

   Then compress your folder into one `zip` file. Submit it to New E3 System with the format **HW3_XXXXXXX.zip** where XXXXXXX is your student ID.

   We **only accept one zip file**, wrong format or naming format cause -10 points to your score (after considering late submission penalty).

3. Late submission lead to score of (original score)*0.85^{days}, for example, if you submit your homework right after the deadline, your get (original score)*0.85 points.

4. If there is anything you are not sure about submission, ask in the forum.