Database System Implementation- CSE 507

Homework 4

Due date: April 12, 2017 12:00 noon

Instructions:

- All submissions must be made through usebackpack site for this course (https://www.usebackpack.com/iiitd/w2017/cse507)
- Only one submission per team would be considered and graded. It would be assumed that all members of the team have participated equally and same score would be given to all members of the team.
- Your submission should have names of all the members of your team.
- Only one submission should be uploaded per team.
- Assumptions made while solving the problem should be clearly stated in the solution. Reasonable ones would be accepted.
- Question 2 is for teams of size 3. This questions will not be graded for teams for size 2 or less.
- As always correctness of the algorithm must be ensured.
- **It is preferred that all algorithms are implemented in the same language.**
- **If you are implementing in C++ then your file extension must be .cc only. No other extension would be accepted.**
- TAs would be quizzing you on your code. You must understand each and every line of your submitted code. Also the implementation specifications mentioned in the questions need to be strictly followed. Failure to adhere to these requirements would result in substantial loss of points.
- **Very Important:** Your code should not have a directory structure. All files (code + dataset + written material for questions) should be present in just one folder. Note that this is absolutely crucial for grading this assignment.

Question 1 (120 points)

For this question, you would be implementing and comparing LRU and MRU page replacement strategies in a simulated environment. Following are the implementation details of the simulation environment and the buffer manager algorithms.

Implementation of main memory buffers for queries:
Buffers are a fixed set of simulated memory locations. You may choose to implement it as a vector of a user defined type “mem location.” During the simulation some of these locations would be holing a valid page which is being used. And others could just be “NULL.” The number of buffers would be give as input.

Page Table:
Page table is a hash map, with key as the “Page Id” and value as the “memory location” in the buffer. The primary goal of the page table is to point to the main memory buffer where a given “page” is located (if at all). If a given “Page Id” is not found in the page table, then it would be considered as “page fault”. And internally, LRU and MRU would have procedures to address these faults. Note that both LRU and MRU would evict a page from buffer only if there are no free buffers available. Appropriate data structures would have to be maintained for smooth running of LRU and MRU.

Simulated Dataset and queries:
For this assignment assume that your database has following three tables: (a) Employee, (b) Department and (c) Project. The tables given below show their statistics.

Table 1: Table Statistics

Table Name	# records	#disk blocks
Employee	10000	1000
Department	1000	100
Project	5000	500

Layout of tables:

(a) Assume that blocks corresponding to Employee, Department and Project tables are laid out sequentially one after another. This means first block of Employee table is located at the physical address "page 1," and first block of the department is located at the physical address "page 1001."

Simulated Queries:

For this assignment, you are required to simulate the page reference behavior of following four queries.

1) Block nested loop join between Department and Employee with Department as the inner loop.
2) Block nested loop join between Department and Employee with Employee as the inner loop.
3) Block nested loop join between Department and Project with Department as the inner loop.
4) Block nested loop join between Department and Project with Project as the inner loop.

Note that you need not fully implement these query processing algorithms. You just need to implement "a skeleton structure" which would have the same page references as the full version. After implementing these "skeletons" run them on the previously mentioned simulated dataset and record the pages referenced in the order in which they would requested by the actual algorithm. You would have to implement a "running" algorithm which reads these page reference patterns. By reading these reference strings, you are implicitly simulating an execution of the block nested loop join. When you read a pageid in the reference pattern, you request the system to arrange for that particular pageid. If the page is present in the main memory, then its reference statistics are updated, otherwise LRU (or MRU) would come in to handle the page replacement.

Please use the following skeleton code for generating the reference strings:

Here we join relation R and S, R is the outer relation and S is the inner relation. Note the comments mentioned in the code, where directs the place of the page references in the reference string.

```
For each block in relation R (Ri's, where i is between 1 and #blocks in R)
    Ri  =  get_next_block(R)                      //Append address of Ri in the reference pattern string
    For each block in relation S (Si's, where i is between 1 and #blocks in S)
        Si  =  get_next_block(S)                  //Append address of Si in the reference pattern string
        For each tuple in Ri  (Tuple_r's, where  r = 1 to #tuples in each block of R)
            Tuple_r = get_next_rec(Ri)            //Append address of Ri in the reference pattern string
            For each tuple in Si  (Tuple_s's, were s = 1 to #tuples in each block of S)
                Tuple_s = get_next_rec(Si)        //Append address of Si in the reference pattern string
            End For
        End For
    End For
End For
```

Experimental Design:

Variable Parameter:

#buffers available for queries = 10, 20, 50, 75, 100, 200, 500, 1000, 1500, 2000, 2500 Space for output buffer is separate and is not counted in these.

Comparison Metric to be recorded:

Measure the total number of page faults encountered by MRU and LRU algorithm for the previously mentioned queries on Employee, Department and Project table.

Testing: TAs would test your code for a small test case with just few buffers and small files. The goal of this testing would be to ensure if you are getting reasonable number of page faults or not. Make sure you code is structured accordingly.

Deliverables for this question:

(a) Code for MRU and LRU
(b) Skeleton code for queries
(c) Dataset (page references for each of the queries)
(d) Final results of the experiments
(e) A brief explanation of trends

Question 2 (30 points, extra question for teams of size 3): For this you need to simulate the behavior of domain separation technique of page replacement. In case the inner relation fits in the main memory completely, then reserve that many buffers for the inner relation and leave rest for the outer relation. Otherwise, give as many buffers as possible to the outer relation while maintaining atleast 1 buffer for the inner relation. Use LRU page replacement inside each domain and repeat the experiment conducted in Question1.

Deliverables for this question:

(a) Code implemented domain separation technique
(b) Final results of the experiments
(c) A brief explanation of trends