Database System Implementation- CSL 520/CSL 620

Homework 2: Total Weightage 13%

Due date: Oct 7 2018 11:55 pm

Instructions:

- 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.
- Any assumptions made while solving the problem should be clearly stated in the solution.
- 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.**
- 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.
- You don't have to use any database system (e.g., PostGRE SQL, Oracle, etc.) for this assignment.

Question 1 (100 points)

For this question you need to need to implement Bitmap and Bit slice indexes on a large file and conduct experiments to evaluate their performance as some experimental parameters are varied. Following are some specifications to be followed in this assignment. Over and above these specifications your design logic should follow the core principles of the Bitmap and Bit slice indexes and any assumptions made should be reasonable.

Dataset creation (and simulating storage on a disk)

For this question, you would have to create a synthetic table (simulating sales records of department stores) containing **10 Lakh records**. Each record in this file contains four fields: (1) Transaction ID (an integer), (2) Transaction sale amount (an integer), (3) Customer name (string) and, (4) category of item. Transaction ID is an integer to identify each transaction uniquely in the dataset. Transaction sale amount is a random integer between 1 and a parameter **theta** (to be varied in experiments). Customer name is a random 3 letter string. You can model this as a character array of length 3. Category of the item is a random integer between 1 --1500 (uniformly distributed across the table). After creating this dataset, you need to simulate its storage on a disk. Define a disk block as a file which can store only 300 records of the synthetic. Assume unspanned organization i.e., records are not allowed to span across two disk block. Following this store your entire synthetic table as collection of these "disk blocks." Each disk block (simulated as a file) should have a unique name, for that you can name them as 1, 2, 3, …, etc. Basically, your original synthetic sales table would be stored as a series of files. The first disk block (file) would store Row 1 -- Row 300; second disk block (second file) would store Row 301 – Row 600. For sake of simplicity use text files for simulating the disk blocks. Also note that each disk block should have an entry at its end which stores the files name of the next disk block for the file. Additionally, in your code you should only store the file name of the first disk block of the synthetic sales table.

Bitmap Index creation:

Create a bit map index on the transaction sale amount. For bit map index, implement both RowID representation and Bit array representation. In RowID representation, we will store the Transaction IDs where that particular value of sale amount is present. Whereas in Bit array representation you need to create an array of length **10 Lakh** (all initialized to zero) and then switch the values in the array to 1 for all transaction IDs which have that particular value of sale amount. Similar to your synthetic table,

you need to store the bitmap index also on disk as disk blocks (similar to previously mentioned logic). In case of RowID representation, your file simulating a disk block should store only 1000RowIDs. In case of bit array representation, assume that each file simulating a disk block can store 32000 "1"s or "0"s.

In addition, given that this bit map index is likely to have many unique values (one vector/rowed representation for each unique sale amount), construct and secondary index on the unique values present in the bit map index. Basically, this secondary index is an associative array with key as the unique sale amount and value as the file name of the first block of the bit vector or RowID representation. For sake of simplicity you may implement this secondary index using a hash_map data structure and keep that in your main memory as the program is running.

Bit slice Index creation:

Create a bit slice index on the transaction sale amount. This would be of 16 bits, which means each the sale amount is represented as a 16-bit integer. Similar to your synthetic table, you need to store the bit slice index also on disk as disk blocks (similar to previously mentioned logic). Assume that each file simulating a disk block can store 32000 "1"s or "0"s. Construct secondary index on the significant bits present in the bit slice index. Basically, this secondary index is an associative array with key as the significant bit and value as the file name of the first block of the bit array (stored as a series of files) . For sake of simplicity you may implement this secondary index also using a hash_map data structure and keep that in your main memory as the program is running.

Query to be studied:

For this question, you would be studying the performance of the previously created indexes for the following query:

Query 1: Select SUM(sale amount) From SALES_TABLE Where <condition>

Assume that the where clause has already been evaluated and given as a bit vector $\mathbf{B_f}$ **(of length 10 Lakh).** In experiments, we will be creating random B_f and evaluating the cost of the different query plans. This bit vector $\mathbf{B_f}$ would be there in the main memory as the program is running. Your code should have a logic which can translate a given RowID to the appropriate filename (aka simulated disk block in sales table) and the entry number instead of doing a linear scan of file.

Experiment 1-a:

Create a random bit array $\mathbf{B_f}$ **(of length 10 Lakh)**. This bit array should have about 1 Lakh "1"s distributed randomly throughout the length of the $\mathbf{B_f}$**.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 3000.**

Experiment 1-b:

Create a random bit array $\mathbf{B_f}$ **(of length 10 Lakh)**. This bit array should have about 1 Lakh "1"s distributed randomly throughout the length of the $\mathbf{B_f}$**.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 50.**

Experiment 2-a:

Create a random bit array $\mathbf{B_f}$ **(of length 10 Lakh)**. This bit array should have about 10,000 "1"s distributed randomly throughout the length of the $\mathbf{B_f}$**.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 3000.**

Experiment 2-b:

Create a random bit array **B_f (of length 10 Lakh)**. This bit array should have about 10,000 "1"s distributed randomly throughout the length of the **B_f.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 50.**

Experiment 3-a:

Create a random bit array **B_f (of length 10 Lakh)**. This bit array should have about 100 "1"s distributed randomly throughout the length of the **B_f.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 3000.**

Experiment 3-b:

Create a random bit array **B_f (of length 10 Lakh)**. This bit array should have about 100 "1"s distributed randomly throughout the length of the **B_f.** Evaluate Query 1 using the following option (a) no-index, (b) Bitmap index with RowID representation, (c) Bitmap index with bit array representation and, (d) Bit-slice index. Report the number of disk blocks (present as files in your implementation) read for each of the mentioned (a), (b), (c) and (d) options. For this experiment set **theta = 50.**

Things to be submitted for this question:

(a) Code implemented for Bitmap index, Bit slice index and code for creating synthetic sales table (and storing it in simulated disk blocks (files)). Code should have proper documentation. Note that if the TA cannot understand your code, you would not be receiving full score.

(b) Data for the recorded in Experiments 1-a, 1-b, 2-a, 2-b, 3-a and 3-b.

(c) A brief (non-trivial) description of the trends observed along with their analysis (why do think the observed trend makes sense?). This text should about 200 words.

Important Note for evaluation:

TAs would be using a small dataset to evaluate the correctness of your code. They would give you a small SALES tables with just 10-15 records. **So you should not hard code table sizes and blocking factors into your code.** They should be taken in as parameters.

Question 2 (extra question for teams of size 3) (50 points)

Please contact instructor first, if you are planning to undertake this question. Create a B+ tree on the category column of the table. As expected this column would have duplicates. You can use B+ tree code from the internet for this question.

For this question, you are expected to determine the total cost (#disk blocks read) for each of the following queries.

Queries to be studied:

(a) Select * From SALES_TABLE Where category = 3;

(b) Select * From SALES_TABLE Where category > 3 and category < 100;

(c) Select * From SALES_TABLE Where category > 3 and category < 500;

Things to be submitted for this question:

(1) Code implemented

(2) Data for queries (a), (b) and (c).

(3) A brief (non-trivial) description of the trends observed along with their analysis (why do think the observed trend makes sense?). This text should about 200 words.