CS127 Homework #4

Due: November 4th, 2019 2:59 P.M.

Warmup #1

File storage and organization:

Since the lecture slides on this topic is terse, you may find looking the information up in the textbook easier. We have therefore provided references to the textbook for these questions.

1. What is a sequential access pattern, and why is it faster than a random access pattern? (p. 436)

2. Free lists are not used for variable length records. Why not? (p. 454)

3. Multitable clusterings are beneficial when the clustered relations are selected in a mutual join operation, but not when a single relation of the clustering is selected. Why is this so? (p. 461-2)

4. Column stores are generally preferable to row stores for data analysis queries because data analysis queries generally select only a few attributes. What features of column stores make them more efficient for such queries and why? (see slides)

Warmup #2

Indexing

1. Why can sparse indices only be clustered/primary indices?

2. Given the relation (dept_name, emp_id, salary) sorted by dept_name (with the memory address of each tuple to the left of the tuple):

addr	emp_id	dept_name	sal
0	0	accounting	100000
1	2	actuarial	120000
2	3	engineering	125000
3	1	engineering	132000

   Build a

   a. dense index on dept_name

   b. When is it preferable to use a primay index rather than a secondary index?

Warmup #3

Hashing

1. What is the difference between static and dynamic hashing?
2. What problem does does dynamic hashing address that static hashing does not?
3. Why is it important to keep track of the local depth in extendible hashing?
4. In Linear Hashing, what happens when try to insert into a bucket that is full?

Problem 4 File Storage

Read the following pages on MySQLs and Postgres storage then answer the questions

MySQL (InnoDB)

- In memory structures: https://dev.mysql.com/doc/refman/8.0/en/innodb-in-memory-structures.html
- On disk structures:
  - https://dev.mysql.com/doc/refman/8.0/en/index-btree-hash.htmlhash-index-characteristics
  - https://dev.mysql.com/doc/refman/8.0/en/innodb-index-types.html
  - https://dev.mysql.com/doc/refman/8.0/en/innodb-physical-structure.html
  - https://dev.mysql.com/doc/refman/8.0/en/sorted-index-builds.htm
  - https://dev.mysql.com/doc/internals/en/innodb-page-structure.html

Postgres

- In memory structures: https://github.com/postgres/postgres/tree/master/src/backend/storage/buffer
- On disk structures:
  - https://www.postgresql.org/docs/12/storage-page-layout.html
  - https://github.com/postgres/postgres/tree/master/src/backend/storage/freespace

Answer the following questions as thoroughly as you can. We'll be reading based on completeness in answering these questions. We answer the first question below to give an example of what we are looking for. (10 points per question)

1. InnoDB seems to support generally two different kinds of indexes: B-trees and Hash Indices.
   (a) When should you choose one over the other?
   (b) Does MySQL always use these indices?
2. Describe InnoDBs strategy to organize tuples into pages.
   (a) That is, how does InnoDB file structure or organization of each of its pages? Describe both how tuples are organized within pages and across pages.
   (b) Describe this file layout and compare it with the file layout we discussed in class.
3. InnoDB requires each relation to have a clustered index.
   (a) What is a clustered index?
   (b) Why does InnoDB require this? If a clustered index isn't used, what does InnoDB do?
4. All databases have a buffer in memory.

   (a) Describe MySQLs strategy to manage its buffer.
   (b) What makes it different from LRU? Why do they do this?

5. Describe Postgres strategy to organize tuples into pages.

6. We talked about free-space maps in class.
   (a) Describe Postgres strategy to keep track of free space in its pages.
   (b) What data structure do they use?
   (c) At what granularity does this strategy record free space?
   (d) How do they keep track of memory across pages?
   (e) How do they handle concurrency?

7. Describe how Postgres eviction and replacement of the page buffer. Do they make special considerations for specific types of workloads? Why?

Problem 5 Indexing - Hashing

Given bucket sizes of 2, and an initial hash of 0 bits, hash 2, 3, 5, 8, 12, 17, 19, 21, 27, 31 in the given order and draw a diagram of the final result, showing all relevant information, (bucket lookup table, corresponding hash keys, pointers, buckets, entries, overflow buckets, local and global depth values, next pointers, level, and anything else relevant). (10 points each)

1. the extendible hashing algorithm given in lecture slides.

2. the linear hashing algorithm given in the lecture slides

Problem 6 Query Processing

1. Design sort-based and hash-based algorithms for computing the relational division operation.

2. Let relations r1(A, B, C) and r2(C, D, E) have the following properties: r1 has 20,000 tuples, r2 has 45,000 tuples, 25 tuples of r1 fit on one block, and 30 tuples of r2 fit on one block. Estimate the number of block transfers and seeks required, using each of the following join strategies for r1 1 r2 (5 points each):
   (a) Block nested-loop join.
   (b) Merge join.
   (c) Hash join.

3. Think about the external merge-sort algorithm. External merge sort is likely applicable to in-memory databases data now moves from RAM to CPU Cache instead of disk to RAM. There is some room for optimization here. Describe a possible optimization to the external merge-sort algorithm and explain how it reduces the cost of the algorithm. Some helpful reference for in-memory sorting might be found starting from slide 34 here: https://bit.ly/2o64dbF (10 points)

4. Describe the fundamental structure of a binary tree. Describe the time complexity for a search on the binary tree. Theoretically, this is as good as it gets. Why then does is a binary tree not frequently used as an indexing structure? Why then does Postgres use this data structure for their free-space map (consider what this binary tree is indexing and whether the same bottlenecks apply)? (10 points)

Problem 7 Query Optimization

1. Consider a relation r1(A, C) and r2(A, B), under what conditions are the following queries equivalent. In the queries below, `func` is one of the SQL aggregation functions: `MIN`, `MAX`, `SUM`, `AVG`, and `COUNT`. It might help to convert this to relational algebra first: (10 points)

```
Select A, B, func(C)
    from r1 join r2
    group by A, B;

Select A, B, C
    from r2 join
    (select A, func(C) as C
        from r1
        group by A);
```

   If func is `MIN` or `MAX`, how can the conditions you gave be relaxed?

2. Describe the intuition as to why $E1 \bowtie_\theta (E2 - E3) \equiv (E1 \bowtie_\theta E2) - (E1 \bowtie_\theta E3)$ (5 points)

Problem 8 Student Feedback

Please fill out the CS127 HW4 Student feedback form to help us better help you to prepare for Quiz 2. (5 points Extra credit)