W4111 - Introduction to Databases Section 02, Fall 2019

Overview

This assignment requires written answers to questions. You may have to copy and paste some SQL statements or scripts into the answer document you submit. You may also have to insert diagrams.

There are 20 questions worth 5 points each. A homework assignment contributes 10 points to your final score. Dividing your score on this assignment by 10 yields the points credited to your final score.

Question 1

Question: Suppose you have data that should not be lost on disk failure, and the application is write-intensive. How would you store the data? (Note: Your answer should consider differences between primarily sequential writes and primarily random writes).

Question 2

Question: Both database management systems (DBMS) and operating systems (OS) provide access to files and implement optimizations for file data access. A DBMS provides significantly better performance for data in databases. How does the DBMS do this? What information does it use? How does it optimize access? Provide some examples.

Question 3

Question: Briefly explain CHS addressing and LBA for disks. Which approach is more common and why? Is it possible to convert a CHS address to an LBA. If yes, how?

Question 4

Question: Explain why the allocation of records to blocks affects database-system performance significantly.

Question 5

Question: Give benefits and disadvantages of variable length record management versus fixed length record management

Question 6

Question: Build and draw a B+ tree after inserting the following values. Assume the maximum degree of the B+ tree is 3.

Values: 3, 11, 12, 9, 4, 6, 21, 9, 15, 2

Question 7

Question: Perform the same insertions in the same order for a hash index. Assume that:

1. The size of the hash table is 13.
2. The hash function is simple modulo.
3. The size of a bucket is one entry.
4. The size of each bucket is one value.
5. The index algorithm uses linear probing to resolve conflicts/duplicates.

Question 8

Question: When is it preferable to use a dense index rather than a sparse index? Explain your answer.

Question 9

Question: Since indexes improve search/lookup performance, which not create an index on very combination of columns?

Question 10

Question: Consider the table below. Add indexes that you think are appropriate for the table and explain your choices. You may use MySQL Workbench to add the indexes. Paste the resulting create statement in the answer section.

Choosing indexes is not possible without understanding use cases/access patterns. Define five use cases and the index you define to support the use case. See the answer section for an example.

```
CREATE TABLE IF NOT EXISTS `customers` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `company` VARCHAR(50) NULL DEFAULT NULL,
  `last_name` VARCHAR(50) NULL DEFAULT NULL,
  `first_name` VARCHAR(50) NULL DEFAULT NULL,
  `email_address` VARCHAR(50) NULL DEFAULT NULL,
  `job_title` VARCHAR(50) NULL DEFAULT NULL,
  `business_phone` VARCHAR(25) NULL DEFAULT NULL,
  `home_phone` VARCHAR(25) NULL DEFAULT NULL,
  `mobile_phone` VARCHAR(25) NULL DEFAULT NULL,
  `fax_number` VARCHAR(25) NULL DEFAULT NULL,
  `address` LONGTEXT NULL DEFAULT NULL,
  `city` VARCHAR(50) NULL DEFAULT NULL,
  `state_province` VARCHAR(50) NULL DEFAULT NULL,
  `zip_postal_code` VARCHAR(15) NULL DEFAULT NULL,
  `country_region` VARCHAR(50) NULL DEFAULT NULL)
```

Answer:

Use Case 1: A user wants to be able find a customer(s) by country_region; country_region and state_province; country_region, state_province, city; just by city.

<Your index definitions go here>

Question 11

Question: Assume that:

1. The query processing engine can use four blocks in the buffer pool to hold disk blocks.
2. The records are fixed size, and each block contains 3 records.
3. There are two relations are R and S. Their formats on disk are:

R	(1,R),(2,R),(3,R)	(4,R),(5,R),(6,R)	(7,R),(8,R),(9,R)
S	(11,S),(4,S),(21,S)	(3,S),(31,S),(13,S)	(5,S),(27,S),(23,S)

Explain how a partition hash join would perform a natural join. You should illustrate your explanation using diagrams of the form below for various steps in the processing:

Step N:

Buffer Pool

(1,T1),(4,T1),(7,T1)
(1,R),(2,R),(3,R)
(4,R),(5,R),(6,R)
(7,R),(8,R),(9,R)

Files

R	(1,R),(2,R),(3,R)	(4,R),(5,R),(6,R)	(7,R),(8,R),(9,R)
S	(11,S),(4,S),(21,S)	(3,S),(31,S),(13,S)	(5,S),(27,S),(23,S)
T1	(3,T1),(6,T1),(9,T1)		

The notation (n,X) means the record in file/relation X with value n for the key. Ti is a temporary file/relation that is created during the processing. You will likely have to create more than one temporary files.

Question 12

Question: Give three reasons why a query processing engine might use a sort-merge join instead of a hash join? What are the key differences sort-merge and hash join?

Question 13

Question:

Let r and s be relations with no indices, and assume that the relations are not sorted. Assuming infinite memory, what is the lowest-cost way (in terms of I/O operations) to compute $r \bowtie s$? What is the amount of memory required for this algorithm?

Question 14

Question:

Rewrite/transform the following query into an equivalent query that would be significantly more efficient.

```
select
        people.playerid, people.nameLast, people.throws,
        batting.teamid, batting.yearid, ab, h, rbi
from
        (people join batting using(playerid))
where teamid='BOS' and yearID='1960';
```

Question 15

Question: Suppose that a B+-tree index on (dept_name, building) is available on relation department. (Note: This data comes from the database at https://www.db-book.com/db7/index.html)

What would be the best way to handle the following selection?

$$\sigma_{(building < \text{“Watson”}) \wedge (budget < 55000) \wedge (dept\_name = \text{“Music”})}(department)$$

Question 16

Question: Consider the following relational algebra expression

$$\pi_{a,b,c}(R \bowtie S)$$

This is a project on the result of a natural join on R and S. Assume that column *a* comes from R, column *b* comes from S and that *c* is the join column. Also assume that both R and S have many large columns. Write an equivalent query that will be more efficient, and explain why.

Question 17

Question:

For each of the following isolation levels, give an example of a schedule that respects the specified level of isolation but is not serializable:

a. Read uncommitted

b. Read committed

c. Repeatable read

Question 18

a. Read uncommitted:

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(A)	

In the above schedule, T_2 reads the value of A written by T_1 even before T_1 commits. This schedule is not serializable since T_1 also reads a value written by T_2, resulting in a cycle in the precedence graph.

b. Read committed:

T_1	T_2
lock-S(A)	
read (A)	
unlock(A)	
	lock-X(A)
	write(A)
	unlock(A)
	commit
lock-S(A)	
read(A)	
unlock-S(A)	
commit	

In the above schedule, the first time T_1 reads A, it sees a value of A before it was written by T_2, while the second **read**(A) by T_1 sees the value written by T_2 (which has already committed). The first read results in T_1 preceding T_2, while the second read results in T_2 preceding T_1, and thus the schedule is not serializable.

c. Repeatable read:
Consider the following schedule, where T_1 reads all tuples in r satisfying predicate P; to satisfy repeatable read, it must also share-lock these tuples in a two-phase manner.

T_1	T_2
pred_*read*(*r*, *P*)	
	insert(t)
	write(A)
	commit
read(A)	
commit	

Suppose that the tuple t inserted by T_2 satisfies P; then the insert by T_2 causes T_2 to be serialized after T_1, since T_2 does not see t. However, the final **read**(A) operation of T_1 forces T_2 to precede T_1, causing a cycle in the precedence graph.

Question: Explain the difference between a *serial schedule* and a *serializable schedule.*

Question 19

Question: What are the benefits and disadvantages of strict two phase locking?

Question 20

Question: Outline the no-steal and force buffer management policies.